Svmonitor

Contents

1 What is svmonitor ?

`Svmonitor` is simple helper that lets you run some tests on your systems and set up actions if they fail.

For example, you could use it to check that some of your important programs are still running, or to regularly check that you haven't run out of disk space...

2 Download

Svmonitor is available both as a Pacman[1] package (x86_64 only), or in source form. For source or Pacman installation, please refer to the Generic download and install instructions[2].

- Latest source tarball[3]
- PGP signature[4]

3 The big picture

`Svmonitor` runs *tests*. Tests are just commands that can fail or succeed. Upon failure or success, the program takes *actions*, like sending email or shutting down your server.

Tests and actions are setup in a global configuration file at `/etc/svmonitor-config/svmonitor.conf`. Each test and action gets referred to using a simple name.

Suppose you have set up a test named `check-disk-usage` that checks if you have enough disk space. Then to run it you would just run:

```
$ runtest check-disk-usage
```

This will take care of running the actual commands and taking the configured actions in case of success or failure.

Actions can be configured to run everytime the test fail, or e.g only once after the first failure.

Note that `svmonitor` does *not* schedule the tests: this can be done using another tool such as fcron[5].

4 The configuration file

The configuration file is in `/etc/svmonitor-config/svmonitor.conf`. It describes two things: the *tests* and the *actions* mapped to those tests.

The file's syntax is very simple (INI[6] like). Sections, whose names are between brackets, contain variables, each of which is given line by line using the `varname = value` format. Any line beginning with a `#` is considered a comment and isn't parsed.

4.1 An example configuration

Here is an example file:

```
[action_mail]

action_command = testaction-sendmail root@localhost
```

[1] http://www.archlinux.org/pacman/
[2] http://svasey.org/projects/download-install-doc_en.html
[3] http://svasey.org/public-repo.svasey.org/src/svmonitor.tar.gz
[4] http://svasey.org/public-repo.svasey.org/src/svmonitor.tar.gz.sig
[5] http://fcron.free.fr/
[6] http://en.wikipedia.org/wiki/INI_file

```
[test_fail]

test_command = false
failure_actions = mail

[test_fail-once]

test_command = false
failure_actions_once = mail

[test_fail-once-2]

test_command = cat /home/sebastien/tmp/svmonitor.test
failure_actions_once = mail

[test_fail-only-once]

test_command = false
failure_actions_once_only = mail

[test_success]

test_command = true
success_actions = mail
```

This file defines five tests, named `fail`, `fail-once`, `fail-once-2`, `fail-only-once` and `success`, and one action, named `mail`.

Formally, the file can describe any number of actions and tests, each of which in its own section.

4.2 Actions

Each action should be in its own section. If the action's name is `$name`, it should be in a section named `action_$name`.

An action can be seen as a command taking three arguments: the error code returned by the test, the test's name, and a path to a file providing more details on the test's result. This command is specified by the `action_command` variable.

The `testaction-sendmail` command from the example is provided by the `svmonitor` package for convenience. Such actions are documented in Implemented actions (section 7).

4.3 Tests

Each test should be in its own section. If the test's name is `$name`, it should be in a section named `test_$name`.

Each test is described by seven variables:

- `test_command`: The command to run to execute the test. If the command returns 0, the test is considered as successfull, otherwise as failed. This variable *must* be given for each test.

- `failure_actions`: Names of actions to take in case of failure, separated by spaces. The actions will be executed in the order in which they are given. If this variable is not given, nothing will be done.
- `failure_actions_once`: Actions to take in case of failure *just after a success*. Those actions will be run only when the test failed, and succeeded the last time. For example, assume your test pings your server and you run the test every 5 minutes. You only want a mail the first time the server is noticed to be down, not every 5 minutes afterward !
- `failure_actions_once_only`: This is even more restrictive than `failure_actions_once`: those actions are run the first time the test fails, and *are never run again*, except after you use a special `--reset` option.
- `success_actions`, `success_actions_once`, `success_actions_once_only`: Like `failure_actions`, but in case the test succeeeds.

5 Usage

To run a test, simply call `runtest $testname`, where `$testname` is the name of your test. Two other modes of operation are possible:

- `runtest --info [$testname]`: Print information on the last result of the test and whether it is "reset", i.e whether the `once_only` actions will run. If no test name is given, this is done for all tests.
- `runtest --reset [$testname]`: Reset the test, so that `once_only` action can run again. If no test name is given, this is done for all tests.

6 The test directory

`svmonitor` must keep track of e.g the previous results of the tests it ran. Those results are stored in `/var/lib/svmonitor`. This directory should be owned by `root:svmonitor`, and its permission bits should be `1770`. Users that want to use `svmonitor` must be members of the `svmonitor` group.

7 Implemented actions

For convenience, some common actions commands are implemented in the `svmonitor` package. They are documented below.

7.1 testaction-sendmail

This sends a mail to the email addresses given as first argument, comma-separated.

8 Implemented tests

For convenience, some common test commands are implemented in the `svmonitor` package. They are documented below.

8.1 diskspace-check

This can be used to check that enough disk space is available. The command exits with 0 status if and only if all partitions have enough disk space.

The definition of enough disk space must be provided by the user. Three methods can be used:

8.1.1 Minimum percentage

This is the simplest one: `diskspace-check` exits with an error if the percentage of available disk space (with respect to the total space) is lower than some threshold. To use that method, just do:

```
$ diskspace-check $minpercent
```

for example, for a threshold of 5%, you will run `diskspace-check 5`.

8.1.2 Minimum percentage with minimum size

This is less restrictive than "pure" minimum percentage: the program will exit with an error if the minimum percentage criteria is satisfied *and* the available space is lower than some threshold. This is useful when working with large drives, e.g assume you have a 1TB drive: if it is 95% full, there is still 50GB available, which is a lot.

To use that method, run:

```
$ diskspace-check $minpercent $minsize
```

The size is assumed to be in bytes by default, but you can use all the standard multipliers, e.g 1KB is 1000 *bytes*, 1Kb is 1000 *bits* and 1KiB is 1024 bytes.

8.1.3 Custom method

You can also specify your own program to do the tests: the program should take as argument the used space and total space, in *bits* and return with 0 status if and only if enough disk space is available. To use that method, run:

```
$ diskspace-check $progpath
```

Where `$progpath` is the path to your program.

8.1.4 Options

Some partitions can of course be excluded from the tests. There are two useful options to do that:

- `--exclude-dev=regexp`: if the device path (first column in `df` output) matches the given regular expression[7], the device is excluded from the tests. Note that invalid device paths (like "none") are also excluded by default.
- `--exclude-mount=regexp`: if the device mountpoint (last column in `df` output) matches the given regular expression, the device is excluded from the tests.

[7] `http://docs.python.org/library/re.html#regular-expression-syntax`

8.2 webpage-check

This can be used to check that a given webpage is responding correctly, and hasn't changed. The script uses wget[8] for downloading the page. It takes two arguments:

- The URL to download
- A local version of the content: if the downloaded file is not exactly the same as the local version, this is an error. This argument is optionnal.

In addition, one option, `--wget-opts`, can be used to specify additionnal space-separated options to pass to `wget`.

8.3 daemon-check

This is used to check that some programs are running on your system. The script takes as arguments the name of the programs that should be running, and compare each of them to the last column of a `ps -ef` output. If the two begin the same way, the program is considered as running, otherwise the test exits with an error.

[8] `http://www.gnu.org/software/wget/`